LinuxDesign

数据规范

数据格式

```
{“type”:”1”,”from”:”发送方”,”pwd”:”用户密码”,”to”:[“接收方1”,“接收方2”],”msg”:”发送信息”}
```

- type：发送类型 0 登录，1 发送，2 注册
- from：信息发起人
- pwd：用户密码
- to：信息接收人，可以为数组，表示群发信息
- msg：发送信息内容

使用插件库

- cJSON：C语言解析 json 数据

定义辅助函数

- iUtil：辅助服务端解析转发消息

系统文件列表

1. **server.c** 服务端代码，函数入口
2. **client.c** 客户端代码，函数入口
3. **cJSON.h** C语言下实现 json 数据序列化和解析的方法库
4. **cJSON.c** cJSON.h方法的实现
5. **util.h** 自定义服务端扩展方法，iUtil 类库
6. **util.c** util.h方法实现
7. **ALLUSERS.JSON** 当前系统注册用户的信息，替代数据库功能
8. **PRINTLOG.LOG** 服务端打印日志记录状态

程序启动

服务器端编译

```
gcc server.c cJSON.c util.c -lm -o server
```

- 编译时添加自定义函数库、系统函数库
- cJSON.c util.c 引用外部函数
- -lm 引用内部函数库 lib cmath
- 启动服务器 `./server`

客户端编译

```
gcc client.c cJSON.c -lm -o client
```

- 编译时添加自定义函数库、系统函数库
- cJSON.c 引用外部函数
- -lm 引用内部函数库 lib cmath
- 启动客户端 `./client`

注册登录

登录验证

```
{“type”:”0”,”from”:”user1”,”pwd”: ”pwd1"}
```

- 验证用户是否存在，查询 ALLUSERS.JSON
- 密码是否正确
- 登录时，密码在终端不回显，使用 getpass 函数

```
================================================================================
        欢迎使用即时聊天系统
        指令说明：
                list[好友列表]          send[发送消息]
                login[用户登录]         register[用户注册]
================================================================================
login
> 输入用户帐号：user1
> 输入用户密码：
● 系统通知 收到消息 来自 [ server ] > 用户登陆成功
```

注册用户

```
{“type”:”2”,”from”:”kangkang”,”pwd”: ”kangkang"}
```

- 检测是否已存在用户，查询 ALLUSERS.JSON
- 在服务器注册用户
- 注册时，密码在终端不回显，使用 getpass 函数

```
==========================================================================
        欢迎使用即时聊天系统
        指令说明：
                list[好友列表]          send[发送消息]
                login[用户登录]         register[用户注册]
==========================================================================
login
> 输入用户帐号：kangkang
> 输入用户密码：
● 系统通知 收到消息 来自 [ server ] > 用户名或密码错误
register
> 输入用户帐号：kangkang
> 输入用户密码：
● 系统通知 收到消息 来自 [ server ] > 用户注册成功
login
> 输入用户帐号：kangkang
> 输入用户密码：
● 系统通知 收到消息 来自 [ server ] > 用户登陆成功
```

服务器转发信息

服务器接收消息类型标识

- `type: x` 加载在线好友列表
- `type: 0` 用户登录
- `type: 1` 用户客户端之间发送消息
- `type: 2` 用户注册
- `type: 3, 4, 5, 6, 7, 8, 9` 服务端返回消息的不同类型，详见 `util.c`

接收登录信息

- 接收登录标识 `type : 0`
- 查询用户列表，是否已注册
- 查询在线列表，是否已登录
- 登录用户，更新在线列表

客户端之间转发消息

- 接收转发标识 `type : 1`
- 查询在线列表，接收方不在线则不发送
- 接收方为多个用户时，转发给每个用户
- 在服务端打印日志文件

接收注册信息

- 接收注册标识 `type : 2`
- 查询用户列表，是否账号已注册
- 为注册用户则添加用户列表，并重写用户文件 `ALLUSERS.JSON`

```
[{
        "name": "user1",
        "pwd":  "pwd1"
    }, {
        "name": "user2",
        "pwd":  "pwd2"
    }, {
        "name": "user3",
        "pwd":  "pwd3"
    }, {
        "name": "user4",
        "pwd":  "pwd4"
    }, {
        "name": "user5",
        "pwd":  "pwd5"
    }, {
        "name": "user6",
        "pwd":  "pwd6"
    }, {
        "name": "kangkang",
        "pwd":  "kangkang"
    }]
```

加载在线列表

- 接收列表标识 `type : x`
- 查询在线好友
- 返回好友列表数组

```
list
● 2016/12/29 14:55:23 在线列表 > [ kangkang, user1, user2, user3 ]
```

```
2016/12/29 14:55:23 > 服务器发送 : {"type":"x", "from":"server", "msg":"kangkang,
user1, user2, user3", "time":"2016/12/29 14:55:23"}
```

程序更能截图

一对一发送消息

```
list
● 2016/12/29 14:55:23 在线列表 > [ kangkang, user1, user2, user3 ]
send
> 选择在线好友：user1
> 输入消息内容：你好，我是kagkang，这是我发给 user1 的消息
```

```
login
> 输入用户帐号: user1
> 输入用户密码:
● 系统通知 收到消息 来自 [ server ] > 用户登陆成功
● 2016/12/29 15:00:14 收到消息 来自 [ kangkang ] > 你好，我是kagkang，这是我发给
 user1 的消息
```

一对多发送消息

```
send
> 选择在线好友: user1,user2,user3
> 输入消息内容: 大家好，我是kangkang，我给三个人发送了消息
```

```
login
> 输入用户帐号: user1
> 输入用户密码:
● 系统通知 收到消息 来自 [ server ] > 用户登陆成功
● 2016/12/29 15:00:14 收到消息 来自 [ kangkang ] > 你好，我是kagkang，这是我发给
 user1 的消息
● 2016/12/29 15:02:20 收到消息 来自 [ kangkang ] > 大家好，我是kangkang，我给三
个人发送了消息
```

```
login
> 输入用户帐号: user2
> 输入用户密码:
● 系统通知 收到消息 来自 [ server ] > 用户登陆成功
● 2016/12/29 15:02:20 收到消息 来自 [ kangkang ] > 大家好，我是kangkang，我给三
个人发送了消息
```

```
login
> 输入用户帐号: user3
> 输入用户密码:
● 系统通知 收到消息 来自 [ server ] > 用户登陆成功
● 2016/12/29 15:02:20 收到消息 来自 [ kangkang ] > 大家好，我是kangkang，我给三
个人发送了消息
```
